https://doi.org/10.21289/KSIC.2016.19.4.167 KSIC

운동계획을 위한 입자 군집 최적화를 이용한 시범에 의한 학습의 적응성 향상

Adaptability Improvement of Learning from Demonstration with Particle Swarm Optimization for Motion Planning

김정중[1*], 이주장[2]

Jeong-Jung Kim[1], Ju-Jang Lee[2]

〈Abstract〉

We present a method for improving adaptability of Learning from Demonstration (LfD) strategy by combining the LfD and Particle Swarm Optimization (PSO). A trajectory generated from an LfD is modified with PSO by minimizing a fitness function that considers constraints. Finally, the final trajectory is suitable for a task and adapted for constraints. The effectiveness of the method is shown with a target reaching task with a manipulator in three-dimensional space.

Keywords : Motion planning, Particle swarm optimization, Learning, Manipulator

1. * 교신저자, KIST, (E-mail: rightcore@gmail.com)
2. KAIST, (E-mail: jjlee@ee.kaist.ac.kr)

1. * Corresponding author, KIST
2. KAIST

1. Introduction

Motion planning finds a sequence of actions that move from an initial configuration to a goal configuration. A representative approximation method for solving high dimensional motion planning problem is Rapidly-Exploring Random Tree (RRT) [1], [2]. The algorithm samples an action in a state space, constructs tree structure and finds a path that starts from an initial configuration to a goal configuration. Another approach for motion planning in a high-dimensional space is a Learning from Demonstration (LfD) strategy [3], [4]. In the LfD, trajectories that a robot should follow are extracted from human demonstrations and those are used as data for encoding motions and they are used for a motion planning. The advantage of the LfD method is that it can find a motion planning solution that is suitable to a task in a short time.

A. Ude and et al. [5] suggested LfD method for various general tasks. In [5], trajectories demonstrated by a human are encoded by Dynamic Motion Primitives (DMPs) [6], [7]. The method was extended in [8] for real-time motion planning. Although the method successfully generates a motion when a query point is similar to learned trajectories, it has a limitation when additional constraints such as an obstacle avoidance, a velocity limitation, and a short distance constraints are added. Even though frameworks for the LfD for dealing with constraints were suggested in [9]-[11], they only consider a few numbers of constraints and setting for the constraints is not tractable.

In this paper, we combine LfD and Particle Swarm Optimization [12], [13] for adaptation of trajectory to a task. Although those two methods can be used independently, their advantages can be utilized when those methods are combined. Amounts of modification for a trajectory generated from the LfD is encoded into a parameter and it is optimized with PSO. The final trajectory becomes suitable for a task and adapts for constraints. The proposed method follows similar procedure shown in [15] and improves adaptation performance by using PSO which is a stochastic global optimization method.

The paper is organized as follows. In section 2, combination method of LfD and PSO are provided. In section 3, the simulation setup is addressed and results of the proposed method is analyzed. Finally concluding remarks and discussion are provided in section 4.

2. Combination of Learning from Demonstration and Particle Swarm Optimization

2.1 Generalization of DMPs for LfD

In this section, LfD framework that is used in the paper is briefly summarized. LfD

method proposed by Ude and et al. [5] is applied to proposed method because it shows a good generalization property and have been applied to many real-world applications.

The procedure for the generalization of DMPs consists of two stages. The first stage is a data collection stage for learning and the second stage is a trajectory generalization stage for motion planning. In the data collection stage, trajectories that should be followed are collected from human demonstrations and the set of trajectories consists of the

$$\boldsymbol{Z} = \left\{ \begin{matrix} y_d^k(t_{k,j}), \dot{y}_d^k(t_{k,j}), \ddot{y}_d^k(t_{k,j}), \boldsymbol{q}_k | \\ k = 1, \cdots, M, j = 1, \cdots, T_k \end{matrix} \right\} \quad (1)$$

where $y_d^k(t_{k,j}), \dot{y}_d^k(t_{k,j}), \ddot{y}_d^k(t_{k,j}), \boldsymbol{q}_k, M, T_k$, and $\boldsymbol{q}_k$ are measured positions, velocities, and acceleration on trajectory k, number of examples, number of sampling points on each trajectory, parameters describing a task. Here, the $\boldsymbol{q}_k$ is used as query points and a structure of the DMPs is determined by these parameters. DMPs are specified by the weighting factor $\boldsymbol{w}$, time constant τ, and goal point g and relations between $\boldsymbol{q}$ and $(\boldsymbol{w}, \tau, g)$ are learned with the data set $\boldsymbol{Z}$,

$$G(\boldsymbol{Z}) : q \rightarrow [\boldsymbol{w}^T, \tau, g]^T.$$

Gaussian process regression (GPR) [16] is used for estimating τ and g and Locally Weighted Regression (LWR) [17] is used for estimating the $\boldsymbol{w}$. For a new query point $\boldsymbol{q}$, it is used as an input of GPR and LWR and time constant τ and goal point g are generated from GPR, and weighting factor $\boldsymbol{w}$ is generated from LWR. DMPs are constructed based on the parameters.

In [8], the original generalization of DMP was referred as Raw Trajectories Generalization (RTG) and the modified generalization of DMP was referred as Movement Primitives Generalization (MPG). The DMP parameters τ and g are estimated with GPR and $\boldsymbol{w}$ is estimated with LWR in RTW and τ, g, and $\boldsymbol{w}$ are estimated with GPR in MPG. A DMP is corresponded to each joint and DMPs are used to generate a trajectory. The joint trajectories for each time are generated with the DMPs and the position of the end-effector is obtained by kinematics. In this paper, the MPG is used for generating a trajectory of LfD.

2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population based stochastic optimization method inspired by social behavior in a nature such as bird flocking or fish schooling [12], [13]. The PSO uses a concept called a particle and a swarm. A particles is corresponding to an animal, a bird, and an insect in a herd, a flock, and a swarm respectively. Each particle has own position and velocity and they are randomly initialized in a search space. When number of particles in a swarm is S and each particle has a D dimensional vector at iteration i, positions and the velocities of particles are represented as (2) and (3) respectively.

$$\boldsymbol{X}_i = \{\boldsymbol{X}_i(1), \boldsymbol{X}_i(2), \cdots, \boldsymbol{X}_i(S-1), \boldsymbol{X}_i(S)\} \quad (2)$$

$$\boldsymbol{V}_i = \{\boldsymbol{V}_i(1), \boldsymbol{V}_i(2), \cdots, \boldsymbol{V}_i(S-1), \boldsymbol{V}_i(S)\} \quad (3)$$

where Xi(s) and Vi(s) are D dimensional vectors.

As increase an iteration, the particles cooperate and finally reach a solution by preserving and sharing their previous best position. Particles store their best experience during optimization process and velocity and position of each particle are updated by (4) and (5) respectively.

$$\boldsymbol{V}_{i+1}(s) = \boldsymbol{V}_i(s) + \phi_1(\boldsymbol{p}_{best}(s) - \boldsymbol{X}_i(s)) + \phi_2(\boldsymbol{l}_{best}(s) - \boldsymbol{X}_i(s)) \quad (4)$$

$$\boldsymbol{X}_{i+1}(s) = \boldsymbol{X}_i(s) + \boldsymbol{V}_{i+1}(s) \quad (5)$$

where Vi(s) is a velocity and Xi(s) is a position of a particle at i iteration. And pbest is a previous best position for each particle and lbest is a best position of the local particle obtained so far. ϕ1 and ϕ2 are determined as ϕ1 = rand(0, c1), ϕ2 = rand(0, c2) and c1 is a cognition learning factor and c2 is a social learning factor.

Algorithm 1 Basic PSO for minimization problem

$i = 0$
for each particle $s = 1, \ldots, S$ **do**
  for each particle $d = 1, \ldots, D$ **do**
    $\mathbf{X}_i[d](s) = uniform(X_{min}[d], X_{max}[d])$
  end for
  $\mathbf{p}_{best}(s) = \mathbf{X}_i(s)$
  $C_i(s) = Cost(\mathbf{X}_i(s))$
end for
for iteration $i = 1, \ldots, MaxIteration$ **do**
  for each particle $s = 1, \ldots, S$ **do**
    $\mathbf{V}_{i+1}(s) = \mathbf{V}_i(s) + \varphi_1(\mathbf{p}_{best}(s) - \mathbf{X}_i(s)) + \varphi_2(\mathbf{g}_{best} - \mathbf{X}_i(s))$
    $\mathbf{X}_{i+1}(s) = \mathbf{X}_i(s) + \mathbf{V}_{i+1}(s)$
    $C_{i+1}(s) = Cost(\mathbf{X}_{i+1}(s))$
    if $C_{i+1}(s) < C_i(s)$ **then**
      $\mathbf{p}_{best}(s) = \mathbf{X}_{i+1}(s)$
    end if
  end for
  $g_{index} = \underset{s}{\text{argmin}} C_{i+1}(s), s = 1, \ldots, S$
  $\mathbf{g}_{best} = \mathbf{X}_{i+1}(g_{index})$
end for
return $\mathbf{g}_{best}$

The optimization procedure of the PSO is summarized in Algorithm 1. First, the velocity and position of particles are initialized in the search space. And then each particle is evaluated and positions of particle are updated until the max iteration.

R. A. Krohling [14] suggested velocity update equation based on the Gausssian distribution random number generator in PSO and named it as Gaussian Swarm. It does not require PSO cognition learning factor c1 and social learning factor c2 and converges faster than canonical PSO. The velocity change equation of Gaussian Swarm is (6)

$$\boldsymbol{V}_{i+1}(s) = |randn|(\boldsymbol{p}_{best}(s) - \boldsymbol{X}_i(s)) + |randn|(\boldsymbol{g}_{best}(s) - \boldsymbol{X}_i(s)) \quad (6)$$

where |randn| is a positive random number generated according to the absolute value of the normal distribution, i.e., abs[N(0,1)].

2.3 Trajectory modification vector for Learning from Demonstration

In the case of the time duration between a start point and a goal point is T and sampling time is dt, number of steps of the trajectories is N = T/dt. The amount of trajectories modification for each step is optimized with PSO. When a trajectory generated from LfD is Plfd(t) and modification

amount is Pmod(t), the final trajectory P(t) for an execution is represented as

$$\begin{aligned} \boldsymbol{P}(t) &= \boldsymbol{P}_{lfd}(t) + \boldsymbol{P}_{\mathrm{mod}}(t), \ \text{for } t = dt,\ldots,(N-1)\times dt, \\ \boldsymbol{P}(t) &= \boldsymbol{P}_{lfd}(t), \ \text{for } t = 0 \ \text{ and } \ t = N\times dt. \end{aligned} \tag{7}$$

P(t) can be a trajectories in Cartesian space or in joints space. When n = t/dt is set, above equations are changed to

$$\begin{aligned} \boldsymbol{P}[n] &= \boldsymbol{P}_{lfd}[n] + \boldsymbol{P}_{\mathrm{mod}}[n], \ \text{for } n = 1,\ldots,N-2, \\ \boldsymbol{P}[n] &= \boldsymbol{P}_{lfd}[n], \ \text{for } n = 0 \ \text{ and } n = N-1. \end{aligned} \tag{8}$$

The procedure for adapting the trajectory for constraints are shown in Fig. 1. The optimization algorithm, PSO, optimizes the $\boldsymbol{P}[n]_{mod}$ and the final trajectory $\boldsymbol{P}[n]$ is obtained by combining $\boldsymbol{P}[n]_{lfd}$ and $\boldsymbol{P}[n]_{mod}$.

2.4 Fitness function for optimization

A fitness function to be minimized has a following form.

$$f(\boldsymbol{P}[n]) = \sum_{i=1}^{NC} W_i Const_i(\boldsymbol{P}[n]) \tag{9}$$

where $Const_i(\cdot)$ are functions that represent each constraints, W_i are weighting factors that balance the relative importance of each constraint and NC is the number of constraints.

The first constraint minimizes the modifications to a trajectory and has the form

$$\begin{aligned} Const_{\min mod}(\boldsymbol{P}[n]) &= \sum_{d=1}^{WD} |\boldsymbol{P}^d[n] - \boldsymbol{P}^d_{lfd}[n]| = \sum_{d=1}^{WD} |\boldsymbol{P}^d_{\mathrm{mod}}[n]| \\ \text{for } \ n &= 0,\ldots,N-1. \end{aligned} \tag{10}$$

A trajectory generated from the LfD should be modified as small as possible to make a trajectory that is similar to a demonstration because the demonstrated trajectory is suitable for a task.

The second constraint is for minimum acceleration and has the form

$$\begin{aligned} Const_{\min acc}(\boldsymbol{P}[n]) &= |acc(\boldsymbol{P}[n])| \\ \text{for } \ n &= 1,\ldots,N-2 \end{aligned} \tag{11}$$

where the function $|acc(\boldsymbol{p})|$ is the magnitude of the acceleration for a step and can be calculated with finite differences. The finite difference equation for each dimension is

$$acc(\boldsymbol{P}[n]) = \frac{\boldsymbol{P}[n+1] - 2\boldsymbol{P}[n] + \boldsymbol{P}[n-1]}{dt^2}. \tag{12}$$

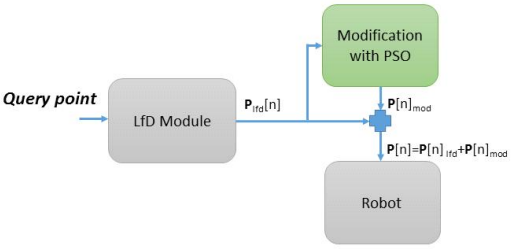

Fig. 1. The procedure for adapting a trajectory for constraints. The final trajectory is combination of a trajectory generated from LfD and a modification amount optimized with PSO.

In this study, dt is set as 1 and $acc(\boldsymbol{P}[n])$ is calculated for each dimension of the workspace. Usually, minimizing the acceleration is important for saving energy and avoiding a sudden change in the motion.

The third constraint is for avoiding obstacles and has a following form

$$Const_{absavo}(\boldsymbol{P}[n]) = \sum_{o=1}^{O} \delta(R[o] + R_{safe}, \|\boldsymbol{Obs}[o] - \boldsymbol{P}[n]\|), \quad \text{for } n = 1,\ldots,N-2 \tag{13}$$

where $\boldsymbol{Obs}[o]$ is the location of the center of the obstacle, $R[o]$ is the radius of the obstacle o, O is the number of obstacles, R_{safe} is the radius for a safe margin and the function $\delta(a_1,a_2)$ is an activation function whose value is 1 when a_2 is less than a_1 and 0 otherwise. In this paper, we assume that obstacles are static and approximated with compositions of spheres.

The final objective function to be optimized is

$$f(\boldsymbol{P}) = \sum_{n=0}^{N-1} \left\{ \begin{array}{l} W_{\min mod} Const_{\min mod}(\boldsymbol{P}[n]) \\ + W_{\min acc} Const_{\min acc}(\boldsymbol{P}[n]) \\ + W_{absavo} Const_{obsavo}(\boldsymbol{P}[n]) \end{array} \right\} \tag{14}$$

The final fitness function is the sum of the fitness value for steps between 0 and $N-1$.

3. Simulations

3.1 Simulation setting

In this section, proposed method is verified by simulations. The simulations are aim for reaching task with an arm-type manipulator which is basic in motion planning. Various tasks such as pick and place task can be utilized based on the task. For implementing a LfD framework, we selected Movement Primitives Generalization (MPG) [8] that can be used for a real-time trajectory generation.

For the simulation, the trajectories that start from an initial position and move to final positions were generated and those were used for learning MPG. The robot being simulated is shown in Fig. 2. The values of the parameters in the figure were $L_1 = 1.7$ cm, $L_2 = 1.7$ cm, $L_3 = 6.7$ cm and $L_4 = 10$ cm. The parameters of the DMP for MPG are $N_{dmp} = 20$, $\beta_z = 1$, $\alpha_z = 4\beta_z$, and $\alpha_x = 0.1$. Sampling time is $dt = 0.01$s, trajectory end time is $T = 0.5$s, and number for the trajectory segment is $N = 50$. We generated 75 minimum jerk trajectories [18] in the joint space that have zero velocity and acceleration at the start point and goal point.

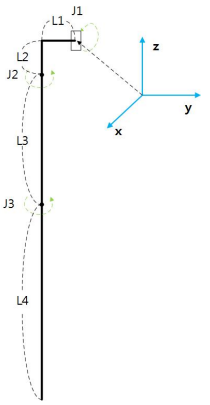

Fig. 2. Kinematic model of the robot arm. The robot has 3 joints and 4 links. The first joint rotates about the y-axis, and the second and third joints rotate about the x-axis.

Final positions in Cartesian coordinate are set as query points. The query point in x axis are located from $12\ cm$ to $16\ cm$ and y and z axis are located from $-4.0\ cm$ to $4.0\ cm$ and the step for each point is set as $2\ cm$. The query points in Cartesian space are shown in Fig. 3.

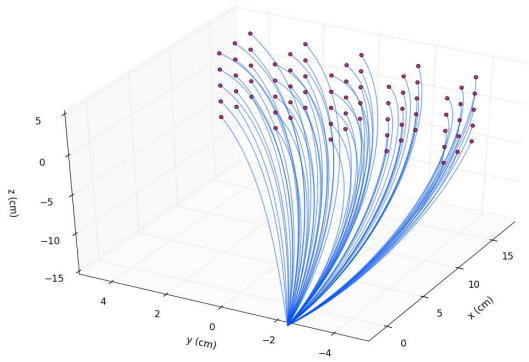

Fig. 3. Query point for learning MPG. Each trajectory starts from an initial position and reach a goal point that is used for a query point. Total number of query point is 75.

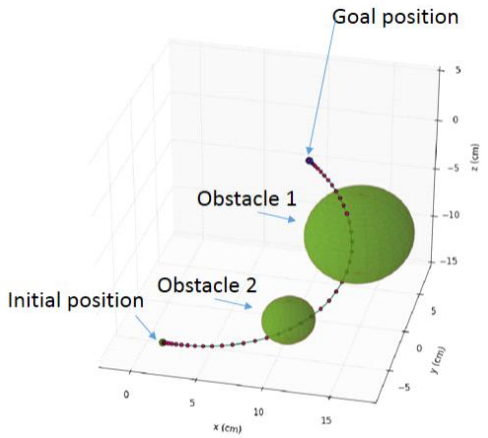

Fig. 4. Trajectory with collision before an adaptation.

The circles in Fig. 3 are query points and there are 75 points. Those trajectories are used for learning the MPG. Obstacles were located at $(15.0, -1.0, -5.0)\ cm$ and $(10.0, -2.0, -14.0)\ cm$ and their radius are $4\ cm$ and $2\ cm$ respectively. When a query point is $(12.0, -4.0, 4.0)\ cm$, a trajectory collides with the obstacles as shown in Fig. 4. In this case, the adaptation should be conducted.

Number of the iteration for PSO was set as 200 and number of particles was set as 100. The initial particles were initialized with normal distribution $N(0,\ 0.01^2)$. The weight parameters were set as $W_{minacc}=W_{minmod}=10$ and $W_{obsavo}=1000$. The safe margin R_{safe} was set as 0.1. In this setting, PSO tries to minimize the fitness function.

3.2 Simulation result

The result of the trajectory adaptation with PSO in three dimensional space is shown in Fig. 5. The trajectory avoided obstacles and reached the goal. The initial trajectory, trajectory modification amount, and final trajectory in each axis are shown in Fig. 6. The steps that violate constraints are optimized and a final trajectory was modified. The fitness value for each iteration is shown in Fig. 7. As shown in Fig. 7, the fitness value was continuously decreased as iteration goes. The initial fitness value was 7935.99 and final fitness value was 299.92. It was reduced for adapting constraints.

4. Conclusion

In this paper, adaptation method for LfD with PSO was proposed. A trajectory

generated from an LfD was modified with PSO. The final trajectory was suitable for task because it was adapted for a constraints. The simulations for generating a trajectory that reaches a goal point in three dimensional space with a robot were conducted. The proposed algorithm successfully modified a trajectory to adapt to constraints and could improve the adaptability of the LfD.

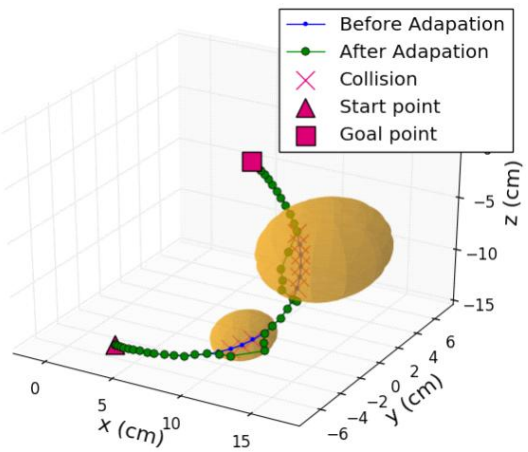

Fig. 5. Trajectory adaptation result. The optimizer modified an orignal trajectory generated from MPG and avoided obstacles.

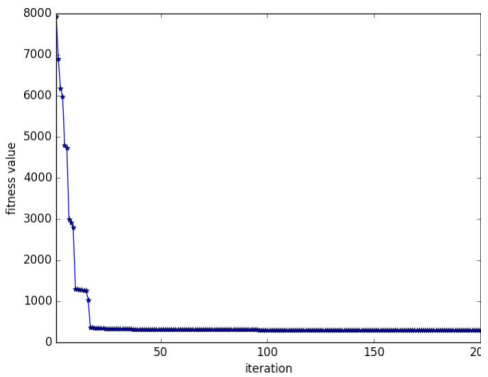

Fig. 7. fitness value for each iteration.

References

[1] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," TR 98-11, Computer Science Dept., Iowa State University, 1998.

[2] S. M. LaValle, "Planning algorithms," Cambridge University Press, 2006.

[3] A. Billard, S. Calinon, R. Dillmann, S.Schaal, "Robot programming by Demonstration," Handbook of Robotics, Springer, pp. 1371-1394, 2008.

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," Robotics and Autonomous Systems, vol. 57, no. 5, pp. 469-483, 2009.

[5] A. Ude, A. Gams, T. Asfour, J. Morimoto, "Task-Specific generalization of discrete and periodic dynamic movement primitives," IEEE Tran. on Robotics, vol. 26, no. 5, 2010.

[6] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in Proc. IEEE Int. Conf. Robot. Autom., pp. 1398-1403, 2002.

[7] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., pp. 958-963, 2002.

[8] D. Forte, A. Gams, J. Morimoto, A. Ude, "On-line motion synthesis and adaptation using a trajectory database," Robotics and Autonomous Systems, vol.

60, pp. 1327-1339, 2012.
[9] S. Calinon, F. Guenter, and A. Billard, “On learning, representing, and generalizing a task in a humanoid robot,” IEEE Trans. on Systems, Man, and Cyber., Part B: Cyber., vol. 37 pp. 286-298, 2007.
[10] S. Calinon, and A. Billard, “A probabilistic programming by demonstration framework handling constraints in joint space and task space,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., pp. 367-372, 2008.
[11] S. Calinon, F. D’halluin, D. G. Caldwell, and A. Billard, 2009, “Handling of multiple constraints and motion alternatives in a robot programming by demonstration framework,” in Proc. IEEE/RAS Int. Conf. Humanoids, pp. 582-588, 2009
[12] J. Kennedy, R. Eberhart, “Particle swarm optimization,” Proc. of the IEEE Int. Conf. on Neural Networks, vol. 4, pp. 1942-1948, 1995.
[13] A. Abraham, H. Guo, and H. Liu, “Swarm intelligence: foundations, perspectives and applications,” Stu. in Compu. Inte., Springer, vol 26, pp. 3-25, 2006.
[14] R. Krohling, “Gaussian swarm: a novel particle swarm optimization algorithm,” IEEE Conf. on Cyber. and Int. Sys., vol. 1, pp. 372-376, Dec. 2004.
[15] J.-J Kim, S.-Y. Park, and J.-J. Lee, “Adaptability improvement of learning from demonstration with sequential quadratic programming for motion planning,” IEEE Conf. on Adv. Int. Mech., pp. 1032-1037, 2015.
[16] C. E. Rasmussen and C. K. I. Williams, “Gaussian processes for machine learning,” MIT Press, 2006.
[17] C. G. Atkeson, A.W. Moore, and S. Schaal, “Locally weighted learning,” Artif. Intell. Rev., vol. 11, pp. 75-113, 1997.
[18] T. Flash, N. Hogan, “The coordination of arm movements: an experimentally confirmed mathematical model,” The Journal of Neuroscience, vol. 5, no. 7, pp. 1688-1703, 1985.

(접수:2016.09.12.,수정:2016.10.10, 개제확정:2016.11.01)